High-Performance Embedded Systems-on-a-Chip

Sanjay Rajopadhye
Computer Science, Colorado State University

Lecture 5: Guibas-Kung-Thompson Array
Outline

- Projects Administrivia
- Manipulating Polyhedra & SRE’s
- Optimal Parenthesization Array [GKT 79]
Projects already “taken”

- Monica Chawathe & Charllie Ross: Interface generation
- Gautam Gupta: Scheduling/Serializing Reductions
- Dae-Kyoo Kim & Eunjee Song: Optimal Prenthesization (protein folding)
- Howard Porter & Stacey Secatch: Custom caches
- Lakshmi Renganarayana: Non-systolic SRE’s
- Jian-Pin Yang & Jhongjun Yang: SOR Kernels
Recap: Parameterized Polyhedra:

\[ \{ z \in \mathbb{Z}^n \mid Qz \geq q \} \]

Dual (alternative, equivalent) representation (vertices/rays):

\[ \{ z \in \mathbb{Z}^n \mid z = a^T V + b^T R; \ a_i, b_i \geq 0; \sum_i a_i = 1 \} \]

Both are useful.

Standard (self-dual) algorithm to transform one to the other.
Preimage by (any) affine function

\[ A(z) \equiv \mathbb{Z}^n \rightarrow \mathbb{Z}^m : z \mapsto Az + a \]

\[ \mathcal{P} \equiv \text{polyhedron} \subseteq \mathbb{Z}^m \]

\[ A^{-1}(\mathcal{P}) \equiv \text{PreImage}(\mathcal{P}, A) \]

\[ \equiv \{ z \in \mathbb{Z}^n \mid A(z) \in \mathcal{P} \} \]
Preimage by (any) affine function

\[ A(z) \equiv \mathbb{Z}^n \rightarrow \mathbb{Z}^m : z \mapsto Az + a \]

\[ \mathcal{P} \equiv \text{polyhedron} \subseteq \mathbb{Z}^m \]

\[ A^{-1}(\mathcal{P}) \equiv \text{PreImage}(\mathcal{P}, A) \]

\[ \equiv \{ z \in \mathbb{Z}^n \mid A(z) \in \mathcal{P} \} \]

by definition \[ = \{ z \in \mathbb{Z}^n \mid Az + a \in \mathcal{P} \} \]
Preimage by (any) affine function

\[ A(z) \equiv \mathbb{Z}^n \rightarrow \mathbb{Z}^m : z \mapsto Az + a \]

\[ \mathcal{P} \equiv \text{polyhedron} \subseteq \mathbb{Z}^m \]

\[ A^{-1}(\mathcal{P}) \equiv \text{PreImage}(\mathcal{P}, A) \]

\[ \equiv \{ z \in \mathbb{Z}^n \mid A(z) \in \mathcal{P} \} \]

by definition \[ \equiv \{ z \in \mathbb{Z}^n \mid Az + a \in \mathcal{P} \} \]

constraint repr. \[ \equiv \{ z \in \mathbb{Z}^n \mid Q(Az + a) \geq q \} \]
Preimage by (any) affine function

\[ A(z) \equiv \mathbb{Z}^n \rightarrow \mathbb{Z}^m : z \mapsto Az + a \]

\[ \mathcal{P} \equiv \text{polyhedron} \subseteq \mathbb{Z}^m \]

\[ A^{-1}(\mathcal{P}) \equiv \text{PreImage}(\mathcal{P}, A) \]

\[ \equiv \{ z \in \mathbb{Z}^n \mid A(z) \in \mathcal{P} \} \]

by definition

\[ \equiv \{ z \in \mathbb{Z}^n \mid Az + a \in \mathcal{P} \} \]

constraint repr.

\[ \equiv \{ z \in \mathbb{Z}^n \mid Q(Az + a) \geq q \} \]

\[ = \{ z \in \mathbb{Z}^n \mid QAz \geq q - Qa \} \]
Preimage by (any) affine function

\[ A(z) \equiv \mathbb{Z}^n \rightarrow \mathbb{Z}^m : z \mapsto Az + a \]

\[ \mathcal{P} \equiv \text{polyhedron } \subseteq \mathbb{Z}^m \]

\[ A^{-1}(\mathcal{P}) \equiv \text{PreImage}(\mathcal{P}, A) \]

by definition

\[ = \{ z \in \mathbb{Z}^n \mid A(z) \in \mathcal{P} \} \]

constraint repr.

\[ = \{ z \in \mathbb{Z}^n \mid QAz + a \in \mathcal{P} \} \]

\[ = \{ z \in \mathbb{Z}^n \mid QAz \geq q - Qa \} \]

Result is a polyhedron with constraints \( \langle QA, q - Qa \rangle \)
Image by (unimodular) affine function

\[ A(z) \equiv \mathbb{Z}^n \to \mathbb{Z}^n : z \mapstoAz + a \]

\[ \mathcal{P} \equiv \text{polyhedron} \subseteq \mathbb{Z}^n \]

\[ A(\mathcal{P}) \equiv \text{Image}(\mathcal{P}, A) \]

\[ \equiv \{ A(z) \in \mathbb{Z}^n \mid z \in \mathcal{P} \} \]
Image by (unimodular) affine function

\[ A(z) \equiv \mathbb{Z}^n \rightarrow \mathbb{Z}^n : z \mapsto Az + a \]

\( \mathcal{P} \equiv \text{polyhedron} \subseteq \mathbb{Z}^n \)

\[ A(\mathcal{P}) \equiv \text{Image}(\mathcal{P}, A) \]

\[ \equiv \{ A(z) \in \mathbb{Z}^n \mid z \in \mathcal{P} \} \]

Result is a polyhedron with vertices \( \langle AV + a \rangle \) and rays \( \langle AR + a \rangle \)
Image by (unimodular) affine function

\[
A(z) \equiv \mathbb{Z}^n \to \mathbb{Z}^n : z \mapsto Az + a
\]

\[
P \equiv \text{polyhedron } \subseteq \mathbb{Z}^n
\]

\[
A(P) \equiv \text{Image}(P, A)
\]

\[
\equiv \{ A(z) \in \mathbb{Z}^n \mid z \in P \}
\]

Result is a polyhedron with vertices \( \langle AV + a \rangle \) and rays \( \langle AR + a \rangle \)

Practical hint: it is also \textit{preimage} by \( A^{-1} \)
Recap: Change of Basis

To obtain an equivalent SRE on applying a CoB transformation by $\mathcal{T}$ to a variable $X$ of an SRE:

\[
X(z) = \left\{ D_i^X : g_i(\ldots Y(f(z)) \ldots) \right\}
\]
Transformation Rules

- Replace each $D_i^X$ by $\mathcal{T}(D_i^X)$, its image by $\mathcal{T}$.
- In the occurrences of a variable on the rhs of the equation for $X$, replace the dependency $f$ by $f \circ \mathcal{T}^{-1}$, the composition\(^a\) of $f$ and $\mathcal{T}^{-1}$.
- In all occurrences of the variable $X$ on the rhs of any equation, replace the dependency $f$ by $\mathcal{T} \circ f$.

\(^a\)Note: function composition is rt associative: $(g \circ h)(z) = g(h(z))$. 
Optimal Parenthesization Array

Compute $C[1, n + 1]$, where, for $1 \leq i < j \leq n + 1$

$$C[i, j] = \begin{cases} i + 1 = j & : f'(i, j) \\ i + 1 < j & : \min_{i < k < j} (C[i, k] + C[k, j] + f(i, j, k)) \end{cases}$$
Optimal Parenthesization Array

Compute $C[1, n + 1]$, where, for $1 \leq i < j \leq n + 1$

$$C[i, j] = \begin{cases} 
  i + 1 = j : & f'(i, j) \\
  i + 1 < j : & \min_{i < k < j} (C[i, k] + C[k, j] + f(i, j, k))
\end{cases}$$

Alternatively (à la Cormen et al) compute $C[1, n]$, where, for $1 \leq i \leq j \leq n$,

$$C[i, j] = \begin{cases} 
  i = j : & 0 \\
  i < j : & \min_{i \leq k < j} (C[i, k] + C[k + 1, j] + f(i, j, k))
\end{cases}$$
Exercise

Prove that the two are equivalent (hint: substitute for the $i = j + 1$ case in the second one and “simplify”).
Exercise

Prove that the two are equivalent (hint: substitute for the $i = j + 1$ case in the second one and “simplify”).

Actual equation (in GKT): $f(i, j, k)$ is independent of $k$: 
Exercise

Prove that the two are equivalent (hint: substitute for the \( i = j + 1 \) case in the second one and "simplify").

Actual equation (in GKT): \( f(i, j, k) \) is independent of \( k \): compute \( C[1, n + 1] \), where, for \( 1 \leq i < j \leq n + 1 \)

\[
C[i, j] = w_{i,j} + \min_{i<k<j} (C[i, k] + C[k, j])
\]
Observations/Questions

- Total volume of computation: $\approx \frac{1}{6} n^3$

- Total I/O volume: $\approx \frac{1}{2} n^2$ (actually $n^2$)

- Triangular array: PE $[i, j]$ computes $C[i, j]$
Observations/Questions

- Total volume of computation: $\approx \frac{1}{6} n^3$
- Total I/O volume: $\approx \frac{1}{2} n^2$ (actually $n^2$)
- Triangular array: PE $[i, j]$ computes $C[i, j]$
- What values are used to compute a given $C[i, j]$?
Observations/Questions

- Total volume of computation: $\approx \frac{1}{6} n^3$

- Total I/O volume: $\approx \frac{1}{2} n^2$ (actually $n^2$)

- Triangular array: PE $[i, j]$ computes $C[i, j]$

- What values are used to compute a given $C[i, j]$?

- Where is a given $C[i, j]$ used?
Observations/Questions

- Total volume of computation: \( \approx \frac{1}{6} n^3 \)
- Total I/O volume: \( \approx \frac{1}{2} n^2 \) (actually \( n^2 \))
- Triangular array: PE \([i, j]\) computes \( C[i, j] \)
- What values are used to compute a given \( C[i, j] \)?
- Where is a given \( C[i, j] \) used?
- With whom is a given \( C[i, j] \) “combined”?
Observations/Questions

- Total volume of computation: $\approx \frac{1}{6} n^3$
- Total I/O volume: $\approx \frac{1}{2} n^2$ (actually $n^2$)
- Triangular array: PE $[i, j]$ computes $C[i, j]$
- What values are used to compute a given $C[i, j]$?
- Where is a given $C[i, j]$ used?
- With whom is a given $C[i, j]$ “combined”?
- Where is it combined?
Observations/Questions

- Total volume of computation: \( \approx \frac{1}{6} n^3 \)
- Total I/O volume: \( \approx \frac{1}{2} n^2 \) (actually \( n^2 \))
- Triangular array: PE \([i, j]\) computes \(C[i, j]\)
- What values are used to compute a given \(C[i, j]\)?
- Where is a given \(C[i, j]\) used?
- With whom is a given \(C[i, j]\) “combined”?
- Where is it combined?
How the array works

- $C[i, j]$ is computed at time $2(j - i)$
How the array works

- $C[i, j]$ is computed at time $2(j - i)$

- It is then sent horizontally and vertically, travelling at a rate of one PE per cycle for exactly $j - i$ more cycles.
How the array works

- \( C[i, j] \) is computed at time \( 2(j - i) \)

- It is then sent horizontally and vertically, travelling at a rate of one PE per cycle for exactly \( j - i \) more cycles.

- Afterwards, it travels at a slower rate of one PE every two cycles.